Efficient Pull-based Mobile Video Streaming leveraging In-Network Functions

Kazuhisa Matsuzono*, Hitoshi Asaeda*, Indukala Naladala† and Thierry Turletti‡
*National Institute of Information and Communications Technology (NICT), Japan. Email: {matsuzono, asaeda}@nict.go.jp
†Department of CSE, National Institute of Technology Karnataka, Surathkal, India. Email: 15co230.indukala@nitk.edu.in
‡Inria, Université Côte d'Azur, France. Email: thierry.turletti@inria.fr

***Abstract*—There has been a considerable increase in the demand for high quality mobile video streaming services, while at the same time, the video traffic volume is expected to grow exponentially. Consequently, maintaining high quality of experience (QoE) and saving network resources are becoming crucial challenges to solve. In this paper, we propose a name-based mobile streaming scheme that allows efficient video content delivery by exploiting a smart pulling mechanism designed for information-centric networks (ICNs). The proposed mechanism enables fast packet loss recovery by leveraging in-network caching and coding. Through an experimental evaluation of our mechanism over an open wireless testbed and the Internet, we demonstrate that the proposed scheme leads to higher QoE levels than classical ICN and TCP-based streaming mechanisms.**

I. Introduction

Mobile video streaming applications and services have been gaining popularity among large players, such as Netflix, Hulu, and Amazon. Many recent studies have highlighted the critical role played by the quality of experience (QoE) perceived by the users, because the revenues for content largely depend on it [1]. From the network operators point of view, due to the rapid growth in video traffic volume, they may require increasing infrastructure investments at a high cost to accommodate large volume of data traffic. It is obvious that 1) achieving high QoE by delivering content faster, and 2) effective utilization of the network and server resources will become more crucial for the success of both the video service providers, and the mobile and fixed-line operators.

There has recently been a strong trend toward the integration of in-network computations and functions with the novel networking architecture to enable efficient content delivery. For instance, Computing in the Network (COIN) [2] in the Internet Research Task Force (IRTF) has been initiated for these novel studies. In-network functions such as in-network caches and flexible video content traffic control have the potential to be embedded into networks as promising approaches. For instance, according to the measurement of the mobile core and backhaul links of Orange France [3], 50% of HTTP requests are cacheable and traffic can be reduced by 60% during the peak hour at least. However, the current major video streaming adopts a connection/host-oriented transport communication (e.g., HTTP on TCP), which makes it difficult to fully exploit the ubiquitous in-network functions due to the connection-oriented nature of application-layer solutions.

Information-Centric Networking (ICN) such as Content-centric networking (CCN) [4] or named-data networking (NDN) [5] can be considered as a promising approach because it adopts a connectionless name-based communication. Indeed, a connectionless hop-by-hop communication makes it easier for video applications to utilize in-network functions such as inherent in-network cache and coding [6], [7] at each node in the path. Furthermore, ICN simplifies the mobility management, where the consumer just needs to send interests without the burden of updating the IP addresses of the consumer and publisher. So, a receiver-driven (pull-based) communication controlled by the consumer requests represents an appealing approach for video streaming applications [9], [10], [11], [12].

Recently, several ICN-based mobile streaming schemes have been proposed [13], [14], [17] and demonstrated great flexibility in mobility management, high resource utilization and enhanced QoE due to significant throughput improvement. However, most of the past research focused on utilizing in-network caches to reduce latency and achieve higher throughput, but not on combining in-network functions to further facilitate video applications. On the other hand, although applying in-network coding was proposed in [6], [7], [8], these studies do not consider an efficient transport for consumer mobility. Furthermore, most of the solutions proposed so far were evaluated through simulations only.

In this paper, envisioning an explosion in mobile video traffic and in the demand for more stable video viewing experience, we propose a new ICN-based transport for mobile video streaming that leverages different in-network functions. Unlike related work which separately considers the transport and in-network functions, we aim to close the gap and develop an efficient transport associated with consumers mobility (e.g., coping with handovers) and in-network functions in a simple yet sophisticated way.

The main contributions of this paper are twofold: 1) We propose an original transport scheme that leverages in-network caching and coding by exploiting request messages controlled by the consumer and the routers in the delivery path, all contributing to further QoE improvement through fast data recovery. 2) By using the actual implementation, we evaluate and demonstrate the efficiency of the proposed scheme through real experiments made on an open wireless testbed and the Internet, and provide along with the paper a detailed

documentation and scripts to reproduce all experiments.

The rest of this paper is organized as follows: Section II provides the background of this work. Section III details the design of the proposed scheme. In Section IV, we evaluate its performance. Section V provides the related work and Section VI concludes the paper.

II. BACKGROUND

A. Basic Communication in ICN

ICN uses two types of packets: interest and data. A publisher splits content objects into data packets, uniquely identified by a name, and cached by the potential routers along the path. A consumer issues interests to trigger data packets delivery on the reverse path followed by the routed interest.

Once receiving an interest, the router executes the following operations: (1) checks if the requested data is cached locally and returns the cached data if cache hits, otherwise, (2) checks if the corresponding entry in pending interest table (PIT) already exists; PIT is a lookup table containing the data name and outgoing Faces (i.e., interfaces) for the data. If so, the router adds the outgoing Face in the PIT and discards the interest. If not, (3) creates a new PIT entry and forwards the interest through the data incoming Faces determined by the adopted forwarding strategy. After receiving the data, the router forwards it through the data outgoing Faces specified by the PIT, stores the forwarded data in its cache if appropriate, and deletes the corresponding PIT entry.

B. Motivation

Although ICN communication provides some advantages, mobile real-time video streaming requires additional sophisticated mechanisms. First, video data should be delivered in time for playback time to avoid playback stalls or rebuffering by promptly recovering lost data in unstable links (e.g., transmission errors or congestion losses in wireless links) and during handovers. Second, interest packet losses on unstable links is important to address because it takes longer time for a consumer to detect possible loss and retransmit the corresponding interest(s). It is also indispensable to avoid intensive interest traffic load in wireless links for effective network resource utilization. In-network functions in ICN have the potential to deal with the issues on improving QoE, but it has not been fully studied and demonstrated how ICN-based mobile streaming effectively utilizes in-network functions to enable efficient and fast data delivery. Thus, there is still room for new designs of ICN-based mobile streaming, and it is worth demonstrating their efficiency through practical experimental results.

III. SYSTEM DESIGN

In this section, we propose a new transport scheme that skillfully uses three types of interests: typical regular interest (RGI), symbolic interest (SMI) [17] and control interest (CNI) according to their characteristics and purposes. Fig. 1 illustrates the system overview.

A. Basic System Model and Assumptions

Video Application: Video producers usually provide real-time video data captured by video cameras (e.g., surveillance cameras), and possibly use multiple-layer codecs like H.264/SVC to prepare a few video qualities. However, compared to single-layer codecs like H.264/AVC, SVC generally requires an additional bandwidth overhead (about 10%) due to a lower compression ratio, and requires longer video encoding/decoding time. In this paper we assume that video servers use single-layer codecs. Each video data is encapsulated in order for the size of ICN packet to be less than the maximum transmission unit (MTU) size when necessary.

To play the target video content, the video application at the consumer side chooses a primary face connected to the wireless link (e.g., LTE), and notifies the mobility transport module of the content name (e.g., /Video.com/Video-A). The player initially buffers some data corresponding to a few seconds of playing time to absorb data acquisition latency.

In-Network Coding Function: In order to add redundancy for fast data recovery, intermediate nodes exploit random linear network coding (RLNC). They generate and forward in a flexible way new coded packets using original data and coded data packets stored in the cache.

In-network coding is applied to each group of k consecutive video data packets, and generates the coded packets using the coefficients of the encoding vector chosen in Galois field $GF(2^8)$. When using coefficients over $GF(2^8)$, the probability of selecting linearly-dependent combinations is negligible. A large value of k could be preferred in terms of increasing robustness against data losses in the network, but this delays decoding for nodes and increases the overhead required to communicate the encoding vector. Thus, a small k value is preferred (e.g., 5).

Coded Data Naming and Acquisition: Each original real-time data has a unique name (content name prefix plus sequence number) in the same manner as with the original CCN or NDN. A unique name is also allocated to coded data using the group indices gid and the encoding vector ev in order to be compatible with the CCN/NDN naming method. For instance, when an $ev = (11, ..., 43)$ is used for $gid = 3$, the coded data name is set to "/Video.com/Video-A/gid=3/ev=(11,...,43)".

B. Consumer Mobility Transport

SMI Transmission: Instead of using RGI for getting *one* data packet specified, the mobility transport module adopts SMI to receive real-time data in an effective way by reducing the interest traffic. The consumer specifies the content name prefix (without the data sequence number) and the SMI lifetime (e.g., 5 s), and then sends the SMI to the connected access point (AP) based on the forwarding information base (FIB) created after finishing the association with the AP. Since the corresponding PIT entries at intermediate nodes are not removed during the SMI lifetime and match any real-time data, this enables continuous data forwarding toward the consumer. For seamless data reception, the consumer

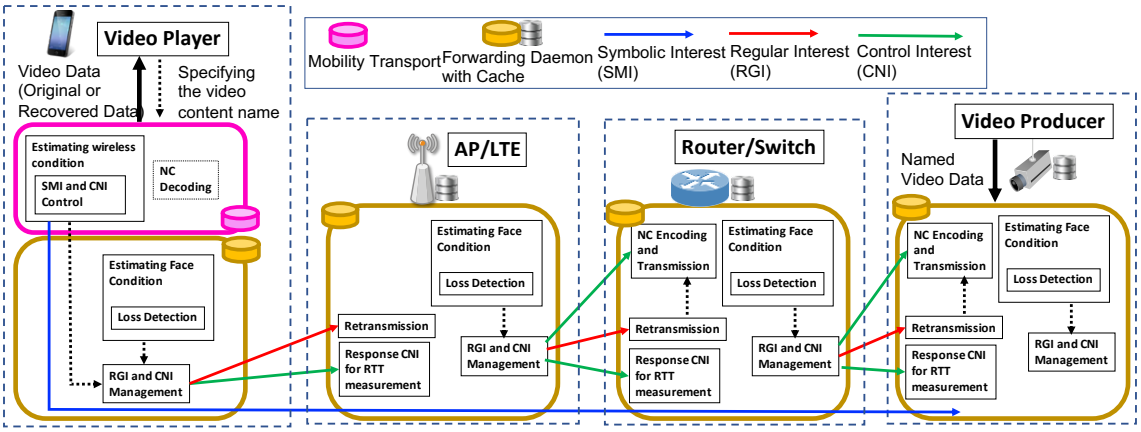


Fig. 1. System overview: the components and related request message (interest) flows.

sends SMIs at regular interval (e.g., 2 s) in order to avoid PITs expiration. However, before and after an handover, the lifetime and sending interval SMI parameters must be carefully determined to (1) avoid unnecessary data traffic transmissions from the previous attached AP, and (2) send as soon as possible a SMI to the newly attached AP. This transport scheme thus adjusts the SMI parameters with the estimated wireless conditions.

Estimating Wireless Condition: The ICN architecture basically maintains a simple relationship with layer-2 as it is layer-2 agnostic. To keep our transport mechanism simple, we adopt a packet-loss based network estimation that does not use layer-2 information. The wireless condition estimation module derives the packet event loss rate of the content flow P_f by monitoring the sequence number of the received data. Then, according to the packet loss observed, it adjusts the SMI parameters of lifetime LT_f and sending interval IT_f (as described next).

Due to the broadcast nature of wireless links, adding redundancy through in-network coding is seldom effective especially in presence of unstable wireless conditions. In such situations, CNI can be sent to disable in-network coding, and RGIs are used to promptly recover lost data by using nearby in-network caches, while SMI reduces the impact of interest losses on data retrieval latency compared to typical streaming that uses RGI only.

Adjusting SMI Parameters: Algorithm 1 describes the adjustments of the SMI parameters. This algorithm attempts to reduce unnecessary data traffic from the previous AP and to promptly receive real-time data from the new AP, while inheriting the SMI benefits of reducing interest traffic and eliminating longer latency caused by interest losses.

The additive increase multiplicative decrease (AIMD) algorithm is adopted for adjusting LT_f, because it is simple and conforms to recv/loss-event based adaptation. In this work, the additive increase factor a and the multiplicative decrease factor b is set to 0.01 s and 0.5 respectively. Based on the determined LT_f, IT_f is basically set to send SMIs at least twice during LT_f. Under unstable wireless conditions, SMI losses may result in stopping real-time data reception. In this case, LT_f should be reduced and its maximum value has to be computed in order for the transmission success probability P_{suc} to exceed the threshold value η_{pit} based on P_f (Line 9). η is set to 0.99 in order to substantially avoid SMI losses. Packet loss events typically occur before handovers, which results in shorter LT_f and IT_f. Such adjustments contribute to avoiding unnecessary data traffic by the previous AP, and to receiving real-time data from the new AP. Finally, the parameters are checked to fall within the range of maximum and minimum values (Line 11). If not, the closest value is chosen. We empirically set the ranges as follows: $0.1 < LT_f(s) < 4.0$, and $0.01 < IT_f(s) < (LT_f - 0.01)/2$.

Algorithm 1 SMI Parameter Adjustment

INPUT:
P_f: packet loss event rate of content flow f,
η_{pit}: threshold of success prob. for updating the AP's PIT,
a: additive increase parameter,
b: multiplicative decrease factor for SMI lifetime.
OUTPUT:
LT_f: SMI lifetime, IT_f: SMI sending interval.

1: **Function** ComputeSMIParams($f, event$)
2: **if** $event = recv$ **then**
3: $LT_F \leftarrow LT_F + a$; $IT_f \leftarrow (LT_f - a)/2$;
4: **end if**
5: **if** $event = loss$ **then**
6: $LT_F \leftarrow LT_F \times b$; $IT_f \leftarrow (LT_f - a)/2$;
7: **end if**
8: **if** $P_{suc} < \eta$ **then**
9: $IT_f \leftarrow Search\_IT_fMax()$;
10: **end if**
11: CheckMaxMin(LT_f, IT_f); return (LT_f, IT_f);

C. Interest/Data Processing with In-Network Functions

Loss Detection and Recovery through In-Network Cache: Each ICN node can detect data losses (on flow f) in the link connected to the upstream node through an in-network

Algorithm 2 In-Network Function Processing

Required:
η_{send}: threshold of success prob. of data transmission,
k: source block length.

1: **At Interest_Packet_Recv**($type, name, gid$)
2: **if** $type$ = RGI **and** get.CacheNum($name, gid$) $\geq k$ **then**
3: send.CodedData($name, gid$);
4: **end if**
5: **if** $type$ = CNI **and** get.CNI_flag() = Coding **then**
6: $CR_{add}^{name} \leftarrow$ get.CodeRate(); /* set minimal redundancy */
7: **end if**
8: **At Data_Packet_Recv**($data\_pkt, recv\_face, name, gid$)
9: /* Data forwarding */
10: **if** get.SentDataNum($name, gid$) $< k$ **then**
11: send.Data($data\_pkt$);
12: **end if**
13: **if** $CR_{add}^{name} \neq 0$ **and** get.CacheNum($name, gid$) $\geq k$ **then**
14: send.CodedData($name, gid, CR_{add}^{name}$);
15: **end if**
16: /* CNI forwarding */
17: $P_{loss} \leftarrow$ get.PktLossRate($data\_pkt, recv\_face$);
18: $D_{allow}^{max/min} \leftarrow$ get.AcceptableLinkDelay($recv\_face$);
19: $CR_{max} \leftarrow$ get.MaxCodeRate($\eta_{send}, P_{loss}, D_{allow}^{max/min}$);
20: **if** $CR_{max} > CR_{prev}$ **then**
21: $cni\_pkt \leftarrow$ create.CNI_pkt(CR_{max});
22: send.CNI(cni_pkt);
23: **end if**

function that monitors the sequence number of received data. After detecting data losses, corresponding RGIs are sent to the upstream node to immediately recover packets lost from the in-network cache of the upstream node. If the video application is too delay-sensitive to use in-network cache for loss recovery, the corresponding RGIs can be not sent to the upstream node [7]. When the upstream node does not cache the original data specified by the RGI but caches the k data packets in the same gid (i.e., including coded data), a re-encoded packet is transmitted as described in Algorithm 2 (Line 3). If a consumer cannot get the corresponding data by the RGI within the average round trip time (RTT) with the upstream node, the RGI is transmitted again. The retransmitted original data by RGI matches the PITs created by SMI and RGI, which leads to duplicated data transmission via the face. Thus, when both PITs coexist, duplicated transmissions can be avoided by adopting longest prefix matching (i.e., RGI's PIT specifying a sequence number takes precedence).

Fast Loss Recovery by In-Network Coding: When applying only-retransmission is not effective, this function attempts to deliver data in a hop-by-hop manner and without video playback stall using minimal redundancy. The downstream node sends CNI and asks for the corresponding upstream node to add the determined minimal redundancy by in-network coding with the cache (Line 21), according to the packet loss rate in the uplink and the estimated acceptable link delay as

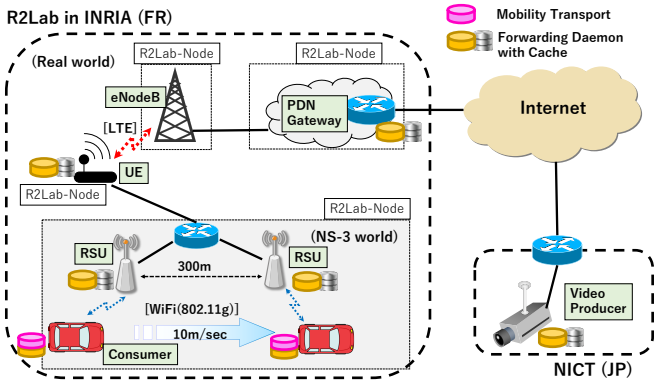
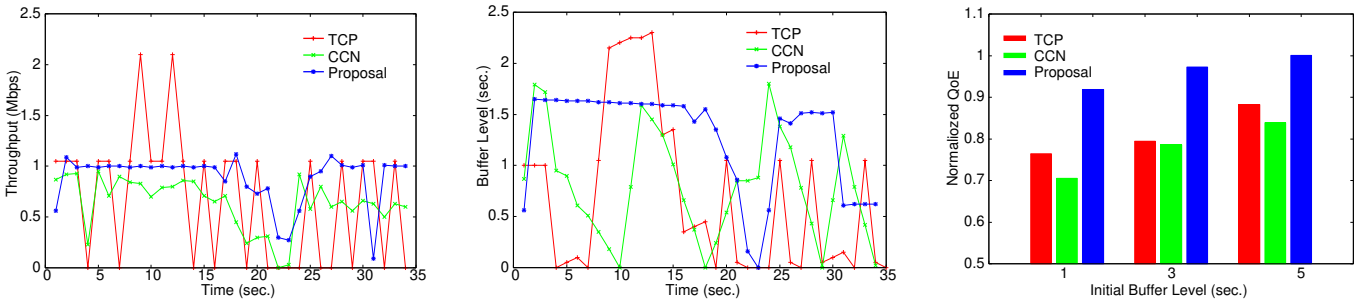


Fig. 2. Experiment environment and topology. In ns-3 world, real implemented programs (i.e., TCP-streaming, Cefore) can be executed by using direct code execution, and the simulated RSUs running on a single R2Lab node can communicate and interact with other real physical node (such as the UE) through tap-bridge devices.

in [7]. Moreover, the added redundant data can be effectively eliminated at the downstream node if necessary: when the downstream node already sends k data in a gid, the received extra data in the gid is not forwarded (Line 10).

The forwarding daemon detects packets lost through the *link sequence number* that the upstream router includes in the optional data header. This mechanism makes it possible to properly derive the packet loss rate P_{loss} for all transmitted data. To allow each node to estimate the acceptable link delay, the mobility transport module and the forwarding daemon perform the following procedure: The mobility transport module specifies the acceptable end-to-end delay from the server to the consumer in a specific field of the SMI optional header, depending on the predefined initial buffer time (e.g., 1 s). Before forwarding SMI and data, each forwarding daemon adds half the values of the maximum and minimum RTT of the uplink to the corresponding fields of the SMI and the data optional header. By referring to the specific fields of SMI and data, each forwarding daemon derives the maximum and minimum link delays of the uplink. Then, based on the derived delays and packet loss rate, each forwarding daemon can calculate the average data transmission success probability in the case of applying retransmission and redundancy. Finally, a minimal redundancy level (i.e., maximum code rate) is determined so as to exceed the threshold success probability $\eta_{send} = 0.995$ (Line 19).

Adding in-network functions of caching and coding generates some processing delay, and thus it may adversely impact on the performance of delay-sensitive video applications like interactive video streaming. Although analyzing and overcoming such added complexity is beyond the scope of this paper, some efficient implementations regarding caching [15] and coding [16] can be utilized.



(a) Throughput (b) Buffer Level (c) Normalized QoE

Fig. 3. Experimental Results.

IV. Evaluation

A. Experiment Setup and Metrics

We implemented the proposed scheme using Cefore [18], [19], an open source software that enables CCN-like communications. To evaluate our mobile video streaming solution, we chose a vehicular scenario in which a mobile user (a video consumer in a vehicle) receives a video from the infrastructure through an hybrid wireless/wired network. Then we compare the performance obtained with two other streaming mechanisms: a basic TCP streaming application and a classical CCN (Cefore-based) streaming application that uses only regular interests (RGI) [9]. Fig. 2 shows the experimentation environment.

We deployed the above scenario using the R2lab wireless testbed [20], an ecosystem that allows running controlled and reproducible experiments with minimal interference. This platform, located at Inria Sophia Antipolis in France, gives remote access to three dozen of nodes (off-the-shelf computers with CPU Intel Core™ i7-2600, 8M Cache at 3.40 GHz, 8GB RAM, 240GB SSD). About half of them are connected to a Software Defined Radio (SDR) device such as USRP B210. R2lab relies on a front-end gateway that allows to control and reach nodes through ssh. To script our scenarios, we used the nepi-ng [21] experiment control tool, which enables concurrent control of the nodes using asyncio[1].

As shown in Figure 2, our scenario also uses the ns-3 network simulator[2] in emulator mode with direct code execution (DCE) [22] on one of the R2lab nodes to emulate two mobile content consumers (cars) running real application software (such as Cefore and TCP streaming). A consumer (in France) is connected to a real LTE network and the Internet through a ns-3 WiFi link in infrastructure mode (between the car and the road side unit RSU) to play a live video service content (published in Japan). At first, the consumer was connected to the left RSU and ran to the right at the speed of 10m/sec. After about 23 s, the low signal strength caused a handover event so as to associate with the right RSU. The bandwidth and delay of wired links were set to 100 Mbps and 1 ms, respectively. Note that consumer adaptation mechanisms used to dynamically select the appropriate quality version are not included in the scenario to focus on the mobility and in-network functions. The publisher thus provides a video content with a single bitrate of about 1.0 Mbps. Data packet sizes of CCN and TCP segments were set to 1460 bytes. We set the RGI interest lifetime to 350 ms, which is larger than the average RTT of 300 ms, in order to avoid PIT expiration without getting the corresponding data. The pipeline size of typical CCN was set to 32, which can achieve a throughput higher than 1.0 Mbps in the absence of packet loss in this experiment.

The scenario shown in Figure 2 is deployed in R2lab using a single nepi-ng script. Basically, the script installs a 4G network (including the core network (EPC), base station (eNodeB) and user equipment (UE)) using OpenAirInterface[3] on 3 R2lab nodes, then it sets up the ns-3 vehicular scenario on another R2lab node, installs Cefore on different locations (shown in Fig. 1) and runs the overall scenario.

As performance metrics, we measured the throughputs, buffer-levels and the QoE levels that are affected by the re-buffering time using [23]: $QoE = \sum_{t=1}^{N} R_t - \lambda \sum_{t=1}^{N} S_t$, where R_t is the bitrate of the t-th segment to be played and S_t is the data amount of stalls for the t-th segment. Based on the settings in [23], we set λ to the video bitrate (i.e., λ=1.0 where the bitrate is represented in the unit of Mbps). We vary the key factor of the initial buffer level from 1 to 5 s (corresponding to play time), since typical live video streaming systems from by service providers assume such an initial-buffering for a few seconds. The results shown in the paper can be reproduced using experimentation scripts provided along with the paper[4] [24].

B. Results

The experimental results are presented in Fig. 3. Fig. 3(b) shows the trace of buffer level at the consumer when the initial buffer level is 1 s, and Fig 3(c) shows the normalized QoE (the ratio to the maximum QoE) when the initial buffer is set to 1, 3 and 5 s.

[1]Asynchronous I/O, URL https://docs.python.org/3/library/asyncio.html.
[2]ns-3 network simulator, see https://www.nsnam.org/.
[3]OAI OpenAirInterface, see https://www.openairinterface.org/.
[4]See https://github.com/fit-r2lab/demo-cefore/blob/master/README.md.

As we can see in Fig. 3(a), the throughput of TCP frequently varies, and often becomes almost 0, mainly due to packet losses in the WiFi network (related to handovers). Moreover, the throughput drastically decreases during handover (from about 21 s) because the connection cannot be maintained. Thus, as shown in Fig. 3(b), the buffer level is often lower than 1 s, which results in rebuffering. In the case of CCN, the throughput is always lower than 1.0 Mbps, because CCN cannot deal with interest and data losses in the network to promptly recover the corresponding data. More specifically, the pipeline control stops issuing a new interest when there are outstanding interests, and interest retransmission is needed to wait for the RGI interest lifetime (350 ms) at least. During the handover, CCN can achieve higher throughput than TCP, since CCN can continuously request the data in a pull-based manner. However, since the overall throughput is low as is the case of TCP, rebuffering often occurs. Accordingly, as shown in Fig. 3(c), the normalized QoE levels obtained for TCP and CCN streaming become less than 0.9 even when the initial buffer level is 5 s. On the other hand, our proposed scheme can achieve higher throughput owing to 1) fast data loss recoveries by the in-network functions and 2) reductions in interest traffic by the SMI control at consumer. This shows that our mechanism can deal with interest and data losses in the network unlike TCP and CCN. Indeed, it is able to maintain a higher QoE level than 0.9 as shown in Fig 3(c).

V. Related Work

Kim et al. [13] argued that data losses during consumer handovers can be effectively recovered by using a simple interest retransmissions at consumer side. However, the performance regarding data retrieval latency is largely affected by the parameter of the loss detection timer, which depends on the interest lifetime parameter of the PIT entry (generally a few seconds). On the other hand, Schneider et al. [14] proposed a forwarding strategy suitable for mobile nodes with multiple interfaces such as WiFi and LTE. They showed that it can improve QoE by dynamically selecting the interfaces to use with the interface probing mechanism for the network estimation. However, since ICN is basically layer-2 agnostic and applicable to any wireless devices, an accurate estimation of the wireless conditions is quite difficult in practice (especially during frequent handovers). Unlike these related studies, we attempt to improve the streaming performance by devising how to use and control interests in the path in order to effectively utilize ICN built-in in-network functions.

Carofiglio et al. [25] proposed an in-network loss detection and recovery mechanism to cope with interest losses at the AP, which inspired us to fully investigate how to utilize in-network functions by controlling interests. In this scheme, the AP detects interest losses by monitoring the sequences of receiving RGIs, and notify the consumer which interests were lost in order to quickly retransmit the interests. On the other hand, our proposed scheme can inherently reduce interest traffic rate and avoid data losses caused by interest losses to conform to real-time mobile video streaming.

Our previous work, NMRTS [17] only focused on how to use interest packets to reduce data retrieval latency and support effective handover for mobile video streaming by suppressing unnecessary interest and data traffic. On the other hand, applying random linear coding as an in-network function was proposed in [6], [7], [8]. These studies did not take into account consumers mobility or data losses in wireless links, but showed that in-network coding can decrease data retrieval latency and enable faster recovery of lost data by exploiting coded data with a specific name. In this work, we consider both how to use interests for effective video data retrieval and in-network coding for fast data recovery.

VI. Conclusion

We proposed an ICN-based mobile video streaming that adjusts the symbolic interest parameters to efficiently retrieve real-time data at consumers, while the intermediate nodes try to recover data losses using in-network cache and coding. We evaluated the proposed scheme by experimentation using a vehicular network scenario involving WiFi, LTE and the Internet, and compared the performance obtained with a typical CCN-streaming and a basic TCP-based streaming mechanism. The results demonstrate that it achieves higher throughput and QoE level. In future we plan to thoroughly evaluate its performance in presence of unstable wireless links and for different handover situations by comparison with some TCP protocols specialized for wireless link. Furthermore, we will add a video quality adaptation mechanism to make a performance evaluation using video content encoded into multiple quality levels.

Acknowledgment

This work was supported by the bilateral program of the Japanese Society for the Promotion of Science (JSPS), by JSPS KAKENHI Grant Number JP19K20264, by the French government through the UCA JEDI (ANR-15-IDEX-01) and by the UHDon5G Associated Team between NICT and Inria.

References

[1] F. Dobrian, et al., "Understanding the impact of video quality on user engagement," Proc. ACM SIGCOMM, Aug. 2011, pp. 362-373.
[2] Computing in the Network (COIN), https://trac.ietf.org/trac/irtf/wiki/coin, accessed July 22, 2019.
[3] G. Carofiglio, M. Gallo, L. Muscariello, and D. Perino, "Scalable mobile backhauling via information-centric networking," Proc. IEEE LANMAN, Apr. 2015, pp. 1–6.
[4] V. Jacobson, et al., "Networking named content," Proc. ACM CoNEXT, Dec. 2009, pp. 1–12.
[5] L. Zhang, et al., "Named data networking," ACM Comput. Commun. Rev., vol. 44, no. 3, 2014, pp. 66–73.
[6] J. Saltarin, E. Bourtsoulatze, N. Thomos, and T. Braun, "Adaptive Video Streaming with Network Coding Enabled Named Data Networking," IEEE Trans., on Multimedia, vol. 19, no. 10, 2017, pp. 2182–2196.
[7] K. Matsuzono, H. Asaeda, and T. Turletti, "Low latency low loss streaming using in-network coding and caching," Proc. IEEE INFOCOM'17, Atlanta, United States, May 2017, pp. 1–9.
[8] E. Bourtsoulatze, et al., "Content-aware delivery of scalable video in network coding enabled named data networks," IEEE Trans. on Multimedia, vol. 20, no. 6, June 2018, pp. 1561–1575.
[9] V. Jacobson, et al., "VoCCN: Voice-over content centric networks," Proc. ACM ReArch Workshop, Dec. 2009, pp. 1–6.

[10] Z. Zhu, et al., “ACT: Audio conference tool over named data networking,” Proc. ACM Workshop on ICN, Aug. 2011, pp. 68–73.
[11] S. Lederer, C. Mueller, C. Timmerer, and H. Hellwagnet, “Adaptive multimedia streaming in information-centric networks,” Proc. IEEE Network Magazine, vol. 28, no. 6, 2014, pp. 91–96.
[12] D. Posch, B. Rainer, and H. Hellwagner, “SAF: Stochastic Adaptive Forwarding in Named Data Networking,” IEEE Trans., on Networking, vol. 25, no. 2, April 2017, pp. 1089–1102.
[13] D. H. Kim et al., “Mobility support in content centric networks,” Proc. ACM ICN Wksp., Aug. 2012, pp. 13–18.
[14] K. M. Schneider and U. R. Krieger, “Beyond network selection: Exploiting access network heterogeneity with named data networking,” Proc. ACM ICN, Nov. 2015, pp. 137–146.
[15] Y. Thomas, G. Xylomenos, C. Tsilopous, and G.C. Polyzos, “Object-oriented Packet Caching for ICN,” Proc. ACM ICN, 2015, pp. 89-98.
[16] M. Pedersen, J. Heide, P. Vingelmann, and F. Fitzek, “Network coding over the $2^{32} - 5$ prime field,” Proc. IEEE ICC, 2013, pp. 2922–2927.
[17] K. Matsuzono and H. Asaeda, “NMRTS: Content Name-Based Mobile Real-Time Streaming,” IEEE Communications Magazine, vol.54, no.8, pp.92–98, Aug. 2016.
[18] H. Asaeda, A. Ooka, K. Matsuzono, and R. Li, ”Cefore: Software Platform Enabling Content-Centric Networking and Beyond,” IEICE Transactions on Communications, vol. E102-B, no. 9, Sep. 2019.
[19] ”Cefore,” Available at: https://cefore.net, accessed July 22, 2019.
[20] “R2Lab: Open Wireless Testbed,” available at: https://r2lab.inria.fr/overview.md/, accessed July 22, 2019.
[21] T. Parmentelat, T. Turletti, W. Dabbous, M.N. Mahfoudi, F. Bronzino, “nepi-ng: an efficient experiment control tool in R2lab,” Proc. ACM WiNTECH, Nov. 2018, pp. 1–8.
[22] H. Tazaki, et al., “Direct code execution: revisiting library OS architecture for reproducible network experiments,” Proc. ACM CoNEXT, Dec. 2013, pp. 217–228.
[23] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, “A control-theoretic approach for dynamic adaptive video streaming over HTTP,” Proc. ACM SIGCOMM, Aug. 2015, pp. 325–338.
[24] “Cefore R2lab experimentation scripts,” available at: https://github.com/fit-r2lab/demo-cefore, accessed May 10, 2019.
[25] G. Carofiglio, et al., “Leveraging ICN in-network control for loss detection and recovery in wireless mobile networks,” Proc. ACM ICN, Sept. 2016, pp. 50–59.